Anime List Design Doc

1. Preamble

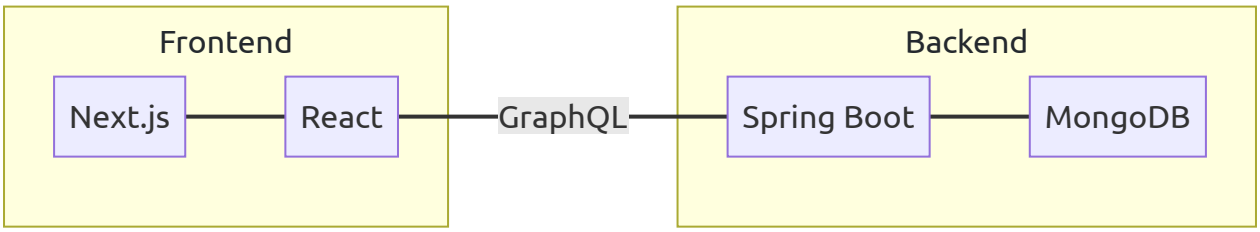
This document will provide an overview of the system architecture for our application.

Although this document is formatted as Markdown, it contains diagrams that will not be rendered by GitHub. So we will maintain a PDF copy of this document here.

2. Background and Motivation

WIP

3. High-Level Architecture



The above diagram shows a high-level diagram of our system architecture.

Our backend consists of two parts: a database for persistence and the main server that handles requests.

For our database, we use MongoDB Atlas. This provides us with a flexible NoSQL database where we can store documents in a JSON format. One primary benefit of this is the ability to model our data within the database in a form that closely matches the way we model the received data in the frontend. Rather than having JSON-shaped data on the frontend and tabular data in the backend, both of them can share a similar structure.

For the main backend server, we use the Spring Boot framework for Java. This allows us to hook into the robust Java ecosystem, giving us libraries to interact with the MongoDB database as well as handle GraphQL requests, which will be discussed a few paragraphs down. In our production deployment, we host this backend server on Heroku.

Our frontend also consists of two parts: the Next.js server that serves the frontend files, and the actual frontend files themselves.

The Next.js server is somewhat of a black box since the implementation is part of the Next.js framework. However, it is still a distinct module in our system architecture, so we will discuss it here. This server is responsible for responding to the client's browser requests and serving the appropriate React frontend files. For example, when the user navigates to `/` the server will respond with the frontend files corresponding to the index page. In our production deployment, this server is hosted on Vercel.

Second, the frontend files that actually run in the user's browser are implemented using React. These files make up the "client" portion of our application, and are responsible for communicating with the backend and displaying the UI of our app.

Finally, as previously mentioned, we use GraphQL to communicate between the frontend and backend! GraphQL is an API specification that acts as an alternative to traditional REST APIs. Rather than making a request to a REST endpoint and receiving data back with a shape determined by the endpoint, GraphQL allows us to traverse a graph of data and specify exactly what objects and fields we want to retrieve. It also allows us to get all of the data we need in a single request, rather than having to make multiple requests to retrieve all of the necessary data. This is because the data is represented as nodes and fields, where the fields can themselves be nodes.

3. Requirements

WIP section